Meta Networks

Tsendsuren Munkhdalai [1] **Hong Yu** [1]

Abstract

Neural networks have been successfully applied in applications with a large amount of labeled data. However, the task of rapid generalization on new concepts with small training data while preserving performances on previously learned ones still presents a significant challenge to neural network models. In this work, we introduce a novel meta learning method, Meta Networks (MetaNet), that learns a meta-level knowledge across tasks and shifts its inductive biases via fast parameterization for rapid generalization. When evaluated on Omniglot and Mini-ImageNet benchmarks, our MetaNet models achieve a near human-level performance and outperform the baseline approaches by up to 6% accuracy. We demonstrate several appealing properties of MetaNet relating to generalization and continual learning.

1. Introduction

Deep neural networks have shown great success in several application domains when a large amount of labeled data is available for training. However, the availability of such large training data has generally been a prerequisite in a majority of learning tasks. Furthermore, the standard deep neural networks lack the ability to continuous learning or incrementally learning new concepts on the fly, without forgetting or corrupting previously learned patterns. In contrast, humans can rapidly learn and generalize from a few examples of the same concept. Humans are also very good at incremental (i.e. continuous) learning. These abilities have been mostly explained by the meta learning (i.e. learning to learn) process in the brain (Harlow, 1949).

Previous work on meta learning has formulated the problem as two-level learning, a slow learning of a meta-level model performing across tasks and a rapid learning of a base-level model acting within each task (Mitchell et al., 1993; Vilalta & Drissi, 2002). The goal of a meta-level learner is to acquire generic knowledge of different tasks. The knowledge can then be transferred to the base-level learner to provide generalization in the context of a single task. The base and meta-level models can be framed in a single learner (Schmidhuber, 1987) or in separate learners (Bengio et al., 1990; Hochreiter et al., 2001).

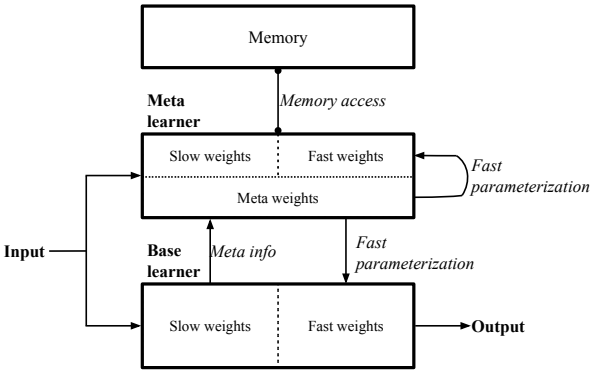

Figure 1. Overall architecture of Meta Networks.

In this work we introduce a meta learning model called MetaNet (for Meta Networks) that supports meta-level continual learning by allowing neural networks to learn and to generalize a new task or concept from a single example on the fly. The overall architecture of MetaNet is shown in Figure 1. MetaNet consists of two main learning components, a base learner and a meta learner, and is equipped with an external memory. Learning occurs at two levels in separate spaces (i.e. meta space and task space). The base learner performs in the input task space whereas the meta learner operates in a task-agnostic meta space. By operating in the abstract meta space, the meta learner supports continual learning and performs meta knowledge acquisition across different tasks. Towards this end, the base learner first analyzes the input task. The base learner then provides the meta learner with a feedback in the form of higher order meta information to explain its own status in the current task space. Based on the meta information,

[1]University of Massachusetts, MA, USA. Correspondence to: Tsendsuren Munkhdalai <tsendsuren.munkhdalai@umassmed.edu>.

Proceedings of the 34^{th} International Conference on Machine Learning, Sydney, Australia, 2017. JMLR: W&CP.

the meta learner rapidly parameterizes both itself and the base learner so that the MetaNet model can recognize the new concepts of the input task. Specifically, the training weights of MetaNet evolve at different time-scales: standard slow weights are updated through a learning algorithm (i.e. REINFORCE), task-level fast weights are updated within the scope of each task, and example-level fast weights are updated for a specific input example. Finally MetaNet equipped with external memory allows for rapid learning and generalization.

Under the MetaNet framework, it is important to define the types of the meta information which can be obtained from the learners. While other representations of meta information are also applicable, we use loss gradients as meta information. MetaNet has two types of loss functions with distinct objectives: a representation (i.e. embedding) loss defined for the good representation learner criteria and a main (task) loss used for the input task objective.

We extensively studied the performance and the characteristics of MetaNet on one-shot supervised learning (SL) problems under several different settings. Our proposed method not only improves the state-of-the-art results on the standard benchmarks, but also shows some interesting properties related to generalization and continual learning.

2. Related Work

what r fast weights?

Our work connects different threads of research in order to model neural architectures for rapid learning and generalization. Rapid learning and generalization refers to a one-shot learning scenario where a learner is introduced to a sequence of tasks, where each task entails multi-class classification with a single or few labeled example per class. A key challenge in this setting is that the classes or concepts vary across the tasks. Due to this, one-shot learning problems have been widely addressed by generative models and metric learning methods. One notable success is reported by a probabilistic programming approach (Lake et al., 2015). They used specific knowledge of how pen strokes are composed to produce characters of different alphabets. Koch (2015) applied Siamese Networks to perform one-shot classification. Recently, Vinyals et al. (2016) unified the training and testing of a one-shot learner under the same procedure and developed an end-to-end differentiable nearest neighbor method for one-shot learning. Santoro et al. (2016) proposed a memory-based approach and trained Neural Turing Machines (Graves et al., 2014) for one-shot learning, although the meta-learner and the one-shot learner in this work are not separable explicitly. The training procedure used by Santoro et al. (2016) adapted the work of Hochreiter et al. (2001) in which they use LSTMs as the meta-level model. More recently an LSTM-based one-shot optimizer was proposed (Ravi & Larochell, 2017). By taking in the loss, the gradient and the parameters of the base learner, the meta optimizer was trained to update the parameters for one-shot classification.

A related line of work focuses on building meta optimizers (Hochreiter et al., 2001; Maclaurin et al., 2015; Andrychowicz et al., 2016; Li & Malik, 2017). As the main interest here is to train an optimization algorithm within the meta learning framework, these efforts have mainly focused on tasks with large datasets. In contrast, with the absence of large datasets, our experimental setup emphasizes the difficulties of optimizing a neural network with a large number of parameters to generalize with limited examples of a new concept. Our work proposes a novel rapid parameterization approach by employing meta information. By following the success of the previous work (Mitchell et al., 1993; Younger et al., 1999; Andrychowicz et al., 2016; Ravi & Larochell, 2017), we study the meta information present in the loss gradient of neural nets. Fast weights and utilizing one neural network to generate parameters for another neural network have previously been studied separately. Hinton & Plaut (1987) suggested the usage of fast weights for rapid learning. Ba et al. (2016) recently used fast weights to replace soft attention mechanism. Fast weights have also been used to implement recurrent nets (Schmidhuber, 1992; 1993a) and self-referential networks (Schmidhuber, 1987; 1993b). These usages of fast weights are well motivated by the fact that synapses have dynamics at many different time-scales (Greengard, 2001).

The approach proposed by Gomez & Schmidhuber (2005) is more closely related to our work. They used recurrent nets to generate fast weights for a single-layer network controller. De Brabandere et al. (2016) used one network to generate slow filter weights for a convolutional neural net. More recently David Ha & Le (2017) generated slow weights for recurrent nets. Our MetaNet generates fast weights at two time-scales by operating in meta space. To integrate the fast weights with the slow weights, we propose a novel layer augmentation approach.

Finally, we note that our MetaNet equipped with an external memory can be seen as a memory augmented neural network (MANN). MANNs have shown promising results on a range of tasks starting from small programming problems (Graves et al., 2014) to large-scale language tasks (Weston et al., 2015; Sukhbaatar et al., 2015; Munkhdalai & Yu, 2017).

3. Meta Networks

MetaNet learns to fast parameterize underlying neural networks for rapid generalizations by processing a higher order meta information, resulting in a flexible AI model that can adapt to a sequence of tasks with possibly distinct in-

Algorithm 1 MetaNet for one-shot supervised learning

Require: Support set $\{x'_i, y'_i\}_{i=1}^N$ and Training set $\{x_i, y_i\}_{i=1}^L$
Require: Base learner b, Dynamic representation learning function u, Fast weight generation functions m and d, and Slow weights $\theta = \{W, Q, Z, G\}$
Require: Layer augmentation scheme
Sample T examples from support set
for $i = 1, T$ **do**
  $\mathcal{L}_i \leftarrow loss_{emb}(u(Q, x'_i), y'_i)$
  $\nabla_i \leftarrow \nabla_Q \mathcal{L}_i$
end for
$Q^* = d(G, \{\nabla\}_{i=1}^T)$
for $i = 1, N$ **do**
  $\mathcal{L}_i \leftarrow loss_{task}(b(W, x'_i), y'_i)$
  $\nabla_i \leftarrow \nabla_W \mathcal{L}_i$
  $W_i^* \leftarrow m(Z, \nabla_i)$
  Store W_i^* in i^{th} position of memory M
  $r'_i = u(Q, Q^*, x'_i)$
  Store r'_i in i^{th} position of index memory R
end for
$\mathcal{L}_{train} = 0$
for $i = 1, L$ **do**
  $r_i = u(Q, Q^*, x_i)$
  $a_i = attention(R, r_i)$
  $W_i^* = softmax(a_i)^\top M$
  $\mathcal{L}_{train} \leftarrow \mathcal{L}_{train} + loss_{task}(b(W, W_i^*, x_i), y_i)$ {Alternatively the base learner can take as input r_i instead of x_i}
end for
Update θ using $\nabla_\theta \mathcal{L}_{train}$

put and output distributions. The model consists of two main learning modules (Figure 1). The meta learner is responsible for fast weight generation by operating across tasks while the base learner performs within each task by capturing the task objective. The generated fast weights are integrated into both base learner and meta learner to shift the inductive bias of the learners. We propose a novel layer augmentation method to integrate the standard slow weights and the task or example specific fast weights in a neural net.

To train MetaNet, we adapt a task formulation procedure by Vinyals et al. (2016). We form a sequence of tasks, where each task consists of a support set $\{x'_i, y'_i\}_{i=1}^N$ and a training set $\{x_i, y_i\}_{i=1}^L$. The class labels are consistent for both support and training sets of the same task, but vary across distinct tasks. Overall the training of MetaNet consists of three main procedures: *acquisition of meta information*, *generation of fast weights* and *optimization of slow weights*, executed collectively by the base and the meta learner. The training of MetaNet is described in Algorithm 1.

To test the model for one-shot SL, we sample another sequence of tasks from a test dataset with unseen classes. Then the model is deployed to classify test examples based on its support set. We assume that we have class labels for the support set during both training and testing. Note that

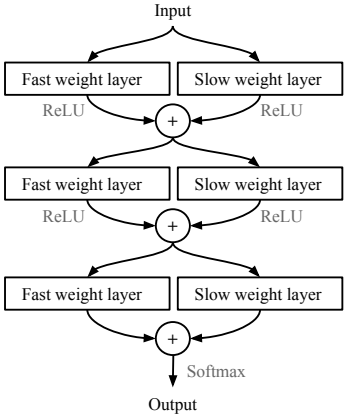

Figure 2. A layer augmented MLP

in one-shot learning setup, the support set contains only single example per class and thus it is cheap to obtain.

3.1. Meta Learner

The meta learner consists of a dynamic representation learning function u and fast weight generation functions m and d. The function u has a representation learning objective and constructs embeddings of inputs in each task space by using task-level fast weights. The weight generation functions m and d are responsible for processing the meta information and generating the example and task-level fast weights.

More specifically, the function m learns the mapping from the loss gradient $\{\nabla_i\}_{i=1}^N$, derived from the base learner b, to fast weights $\{W_i^*\}_{i=1}^N$:

$$W_i^* = m(Z, \nabla_i) \tag{1}$$

where m is a neural network with parameter Z. The fast weights are then stored in a memory $M = \{W_i^*\}_{i=1}^N$. The memory M is indexed with task dependent embeddings $R = \{r'_i\}_{i=1}^N$ of the support examples $\{x'_i\}_{i=1}^N$, obtained by the dynamic representation learning function u.

The representation learning function u is a neural net parameterized by slow weights Q and task-level fast weights Q^*. It uses the representation loss $loss_{emb}$ to capture a representation learning objective and to obtain the gradients as meta information. We generate the fast weights Q^* on a per task basis as follows:

$$\mathcal{L}_i = loss_{emb}(u(Q, x'_i), y'_i) \tag{2}$$

$$\nabla_i = \nabla_Q \mathcal{L}_i \tag{3}$$

$$Q^* = d(G, \{\nabla\}_{i=1}^T) \tag{4}$$

where d denotes a neural net parameterized by G, that accepts variable sized input. First, we sample T examples ($T \leq N$) $\{x'_i, y'_i\}_{i=1}^{T}$ from the support set and obtain the loss gradient as meta information. Then d observes the gradient corresponding to each sampled example and summarizes into the task specific parameters. We use LSTM for d although the order of inputs to d does not matter. Alternatively we can take summation or average of the gradients and use a MLP. However, in our preliminary experiment we observed that the latter results in a poor convergence.

Once the fast weights are generated, the task dependent input representations $\{r'_i\}_{i=1}^{N}$ are computed as:

$$r'_i = u(Q, Q^*, x'_i) \tag{5}$$

where the parameters Q and Q^* are integrated using the layer augmentation method described in Section 3.3.

The loss, $loss_{emb}$ does not need to be the same as the main task loss $loss_{task}$. However, it should be able to capture a representation learning objective. We use cross-entropy loss when the support set has only a single example per class. When there are more than one examples per class available, contrastive loss (Chopra et al., 2005) is a natural choice for $loss_{emb}$ since both positive and negative samples can be formed. In this case, we randomly draw T number of pairs to observe the gradients and the loss is

$$\mathcal{L}_i = loss_{emb}(u(Q, x'_{1,i}), u(Q, x'_{2,i}), l_i) \tag{6}$$

where l_i is auxiliary label:

$$l_i = \begin{cases} 1, & \text{if } y'_{1,i} = y'_{2,i} \\ 0, & \text{otherwise} \end{cases} \tag{7}$$

Once the parameters are stored in the memory M and the memory index R is constructed, the meta learner parameterizes the base learner with the fast weights W_i^*. First it embeds the input x_i in the task space by using the dynamic representation learning network (i.e. Equation 5) and then reads the memory with soft attention:

$$a_i = attention(R, r_i) \tag{8}$$

$$W_i^* = norm(a_i)^\top M \tag{9}$$

where $attention$ calculates similarity between the memory index and the input embedding and we use cosine similarity as $attention$ and $norm$ is a normalization function, for which we use $softmax$.

3.2. Base Learner

The base learner, denoted as b, is a function or a neural net that estimates the main task objective via a task loss $loss_{task}$. However, unlike standard neural nets, b is parameterized by slow weights W and example-level fast weights W^*. The slow weights are updated via a learning algorithm during training whereas the fast weights are generated by the meta learner for every input.

The base learner uses a representation of meta information obtained by using a support set, to provide the meta learner with feedbacks about the new input task. The meta information is derived from the base learner in form of the loss gradient information:

$$\mathcal{L}_i = loss_{task}(b(W, x'_i), y'_i) \tag{10}$$

$$\nabla_i = \nabla_W \mathcal{L}_i \tag{11}$$

Here $\mathcal{L}_i$ is the loss for support examples $\{x'_i, y'_i\}_{i=1}^{N}$. N is the number of support examples in the task set (typically a single instance per class in the one-shot learning setup). ∇_i is the loss gradient with respect to parameters W and is our meta information. Note that the loss function $loss_{task}$ is generic and can take any form, such as a cumulative reward in reinforcement learning. For our one-shot classification setup we use cross-entropy loss. The meta learner takes in the gradient information ∇_i and generates the fast parameters W^* as in Equation 1.

Assuming that the fast weights W_i^* for input x_i are defined, the base learner performs the one-shot classification as:

$$P(\hat{y}_i | x_i, W, W_i^*) = b(W, W_i^*, x_i) \tag{12}$$

where $\hat{y}_i$ is predicted output and $\{x_i\}_{i=1}^{L}$ is an input drawn from the training set $\{x_i, y_i\}_{i=1}^{L}$ for the current task. Alternatively the base learner can take as input the task specific representations $\{r_i\}_{i=1}^{L}$ produced by the dynamic representation learning network, effectively reducing the number of MetaNet parameters and leveraging shared representations. In this case, the base learner is forced to operate in the dynamic task space constructed by u instead of building new representations from the raw inputs $\{x_i\}_{i=1}^{L}$.

During training, given output labels $\{y_i\}_{i=1}^{L}$, we minimize the cross-entropy loss for one-shot SL. The training parameters of MetaNet θ consists of the slow weights W and Q and the meta weights Z and G (i.e. $\theta = \{W, Q, Z, G\}$) and jointly updated via a training algorithm such as back-propagation to minimize the task loss $loss_{task}$ (Equation 12).

In a similar way, as defined in the Equation 2-4, we can also parameterize the base learner with task-level fast weights. An ablation experiment on different variation of MetaNet is reported in Section 4.

3.3. Layer Augmentation

A slow weight layer in the base learner is extended with its corresponding fast weights for rapid generalization. An

Table 1. One-shot accuracy on Omniglot previous split

Model	5-way	10-way	15-way	20-way
Pixel kNN (Kaiser et al., 2017)	41.7	-	-	26.7
Siamese Net (Koch, 2015)	97.3	-	-	88.1
MANN (Santoro et al., 2016)	82.8	-	-	-
Matching Nets (Vinyals et al., 2016)	98.1	-	-	93.8
Neural Statistician (Edwards & Storkey, 2017)	98.1	-	-	93.2
Siamese Net with Memory (Kaiser et al., 2017)	98.4	-	-	95.0
MetaNet-	98.4	98.32	96.68	96.13
MetaNet	**98.95**	**98.67**	**97.11**	**97.0**
MetaNet+	98.45	97.05	96.48	95.08

example of the layer augmentation approach applied to a MLP is shown in Figure 2. The input of an augmented layer is first transformed by both slow and fast weights and then passed through a non-linearity (i.e. $ReLU$) resulting in two separate activation vectors. Finally the activation vectors are aggregated by an element-wise vector addition. For the last $softmax$ layer, we first aggregate two transformed inputs and then normalize for classification output.

Intuitively, the fast and slow weights in the layer augmented neural net can be seen as feature detectors operating in two distinct numeric domains. The application of the non-linearity maps them into the same domain, which is $[0, \infty)$ in the case of $ReLU$ so that the activations can be aggregated and processed further. Our aggregation function here is element-wise sum.

Although it is possible to define the base learner with only fast weights, in our preliminary experiment we found that the integration of both slow and fast weights with the layer augmentation approach is essential in convergence of MetaNet models. A MetaNet model relying on a base leaner with only fast weights were failed to converge and the best performance of this model was reported to be as equal as that of a constant classifier that assigns the same label to every input.

4. Results

We carried out one-shot classification experiments on three datasets: Omniglot, Mini-ImageNet and MNIST. The Omniglot dataset consists of images across 1623 classes with only 20 images per class, from 50 different alphabets (Lake et al., 2015). It also comes with a standard split of 30 training and 20 evaluation alphabets. Following (Santoro et al., 2016), we augmented the training set through 90, 180 and 270 degrees rotations. The images are resized to 28 x 28 pixels for computational efficiency. For the experiment on Mini-ImageNet data, we evaluated on the same class subset provided by Ravi & Larochell (2017). MNIST images were used as out-of-domain data. The training details are described in Appendix A.

4.1. One-shot Learning Test

In this section we will report four groups of benchmark experiments: Omniglot previous split, Mini-ImageNet, MNIST as out-of-domain data and Omniglot standard split.

4.1.1. Omniglot Previous Split

Following the previous setup Vinyals et al. (2016), we split the Omniglot classes into 1200 and 423 classes for training and testing. We performed 5, 10, 15 and 20-way one-shot classification and compared our performance against the state-of-the-art results. We also studied three variations of MetaNet as an ablation experiment in order to show how fast parameterization affects the network dynamics.

In Table 1, we compared the performance of our models with all published models (as baselines). The first group of methods are the previously published models. The next group is MetaNet variations. MetaNet is the main architecture described in Section 3. MetaNet- is a variant without task-level fast weights Q^* in the embedding function u whereas MetaNet+ has additional task-level weights for the base learner in addition to W^*. Our MetaNet model improves the previous best results by 0.5% to 2% accuracy. As the number of classes increases (from 5-way to 20-way classification), overall the performance of the one-shot learners decreases. MetaNet's performance drop is relatively small (around 2%) while the drop for the other models ranges from 3% to 15%. As a result, our model shows an absolute improvement of 2% on 20-way one-shot task.

Comparing different MetaNet variations, the additional task-level weights in the base learner (MetaNet+) did not seem to help and in fact had a negative effect on performance. MetaNet- however performed surprisingly well but still falls behind the MetaNet model as it lacks the dynamic representation learning function. This performance gap increases when we test them in out-of-the domain setting (Appendix B).

Table 2. One-shot accuracy on Mini-ImageNet test set

Model	5-way
Fine-tuning (Ravi & Larochell, 2017)	28.86 ± 0.54
kNN (Ravi & Larochell, 2017)	41.08 ± 0.70
Matching Nets (Vinyals et al., 2016)	43.56 ± 0.84
MetaLearner LSTM (Ravi & Larochell, 2017)	43.44 ± 0.77
MetaNet	**49.21 ± 0.96**

4.1.2. MINI-IMAGENET

The training, dev and testing sets of 64, 16, and 20 ImageNet classes (with 600 examples per class) were provided by Ravi & Larochell (2017). By following Ravi & Larochell (2017), we sampled 15 examples per class for evaluation. By using the dev set, we set an evaluation checkpoint where only if the model performance exceeds the previous best result on random 400 trials produced from the dev set, we apply the model to another 400 trials randomly produced from the testing set and report the average accuracy.

In Table 2, we present the results of the 5-way one-shot evaluation. MetaNet improved the previous result by up to 6% accuracy and obtained the best result.[1]

4.1.3. OMNIGLOT STANDARD SPLIT

Omniglot data comes with a standard split of 30 training alphabets with 964 classes and 20 evaluation alphabets with 659 classes. We trained and tested only the standard MetaNet model in this setup. In order to best match the evaluation protocol of Lake et al. (2015), we form 400 tasks (trials) from the evaluation classes to test the model.

In Table 3, we listed the MetaNet results along with the previous models and human performance. Our MetaNet outperformed the human performance by a slight margin, but underperformed the probabilistic programming approach. However, the performance gap is rather small between these top three baselines. In addition while the probabilistic programming performs slightly better than MetaNet, our model does not rely on any extra prior knowledge about how characters and strokes are composed. Comparing the results on two Omniglot splits in Tables 1 and 3, MetaNet showed decreasing performances on the standard split. The later setup seems to be slightly difficult as the number of classes in the training set is less (1200 vs 964) and test classes are bigger (423 vs 659).

[1]Our code and data will be made available at: https://bitbucket.org/tsendeemts/metanet

4.2. Generalization Test

We conducted a set of experiments to test the generalization of MetaNet from multiple aspects. The first experiment tests whether a MetaNet model trained on an N-way one-shot task could generalize to another K-way task (where $N \neq K$) without actually training on the second task. The second experiment is to test if a meta learner trained for rapid parameterization of a base learner b_{train} could parameterize another base learner b_{eval} during evaluation. The last experimental setup examines whether MetaNet supports meta-level continual learning.

4.2.1. N-WAY TRAINING AND K-WAY TESTING

In this experiment, MetaNet is trained on N-way one-shot classification task and then tested on K-way one-shot tasks. The number of training and test classes are varied (i.e. $N \neq K$). To handle this, we inserted a $softmax$ layer into the base learner during evaluation and then augmented it with the fast weights generated by the meta learner. If the meta learner is generic enough, it should be able to parameterize the new $softmax$ layer on the fly. The new layer weights remained fixed since no parameter update was performed for this layer. The K-way test tasks were formed from the 423 unseen classes in the test set.

The MetaNet models were trained on one of 5, 10, 15 and 20-way one-shot tasks and evaluated on the rest. In Table 4, we summarized the results. As a comparison we also included some results from Table 1, which reports accuracy of N-way train and test setting. The MetaNet model trained on 5-way tasks obtained 93.07% of 20-way test accuracy which is still a closer match to Matching Network and higher than Siamese Net trained 20-way tasks. An interesting finding is that when N is smaller than K, i.e. the model is trained on easier tasks than test ones, we observe a decreasing performance. Conversely the models trained on harder tasks (i.e. $N > K$) achieved increasing performances when tested on the easier tasks and the performance is even higher than the ones that were applied to the tasks with the same level difficulty (i.e. $N = K$). For example, the model skilled on 20-way classification improved the 5-way one-shot baseline by 0.6% showing a ceiling performance in this setting. We also conducted a preliminary experiment on more extreme test-time classification. MetaNet trained on 10-way task achieved around 65% on 100-way one-shot classification task.

This flexibility in MetaNet is crucial because one-shot learning usually involves an online concept identification scenario. Furthermore we can empirically obtain a performance lower or upper bound. Particularly the test performance obtained on the tasks with the same level difficulty that the model was skilled on can be used as a performance lower or an upper bound depending on a scenario under

Table 3. One-shot accuracy on Omniglot standard split

Model	5-way	10-way	15-way	20-way
Human performance (Lake et al., 2015)	-	-	-	95.5
Pixel kNN (Lake et al., 2013)	-	-	-	21.7
Affine model (Lake et al., 2013)	-	-	-	81.8
Deep Boltzmann Machines (Lake et al., 2013)	-	-	-	62.0
Hierarchial Bayesian Program Learning (Lake et al., 2015)	-	-	-	**96.7**
Siamese Net (Koch, 2015)	-	-	-	92.0
MetaNet	98.45	97.32	96.4	95.92

which the model will be deployed in the future. For example, for the MetaNet model that will deployed under the $N > K$ scenario, we can obtain the performance lower bound by testing it on the $N = K$ tasks.

4.2.2. Rapid Parameterization of Fixed Weight Base Learner

We replaced the entire base learner with a new CNN during evaluation. The slow weights of this network remained fixed. The fast weights are generated by the meta learner that is trained to parameterize the old base learner and used to augmented the fixed slow weights.

We tested a small and a large CNN for the base learner. The small CNN has 32 filters and the large CNN has 128 filters. In Figure 3, the test performances of these CNNs are compared. The base learner (target CNN) optimized along within the model performed better than the fixed weight CNNs. The performance difference between these models is large in earlier training iterations. However, as the meta learner sees more one-shot learning trials, the test accuracies of the base learners converge. This results show that MetaNet effectively learns to parameterize a neural net with fixed weights.

4.2.3. Meta-Level Continual Learning

MetaNet operates in two spaces: input problem space and meta (gradient) space. If the meta space is problem independent, MetaNet should support meta-level continual learning or life-long learning. This experiment tests this in the case of the loss gradient.

Table 4. Accuracy of MetaNet trained on N-way and tested on K-way one-shot tasks

	Test			
Train	5-way	10-way	15-way	20-way
5-way	98.95	96.4	93.6	93.07
10-way	99.25	96.87	96.95	96.21
15-way	99.35	98.17	97.11	96.36
20-way	99.55	98.87	97.41	97.0

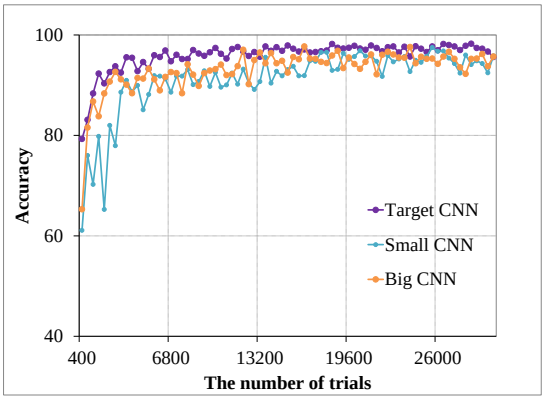

Figure 3. Comparison of the test performances of the base learners on Omniglot 5-way classification.

Following the previous work on catastrophic forgetting in neural networks (Srivastava et al., 2013; Goodfellow et al., 2014; Kirkpatrick et al., 2016), we formulated two problems in a sequential manner. We first trained and tested the model on the Omniglot sets and then we switched and continued training on the MNIST data. After training on a number of MNIST one-shot tasks, we re-evaluated the model on the same Omniglot test set and compare performance. A decrease in performance here indicates that the meta weights Z and G of the neural nets m and d are prone to catastrophic forgetting and the model therefore does not support continual learning. On the other hand, an increase in performance indicates that MetaNet supports reverse transfer learning and continual learning.

We allocated separate parameters for the weights W and Q when we switched the problems so the only meta weights were updated. We used two three-layer MLPs with 64 hidden units as the embedding function and the base learner. The MNIST image and classes were augmented by randomly permuting the pixels. We created 50 different random shuffles and thus the training set for the second one-shot problem consisted of 500 classes. We conducted multiple runs and increased the MNIST training trials by multiples of 400 (i.e. 400, 800, 1200...) in each run giving

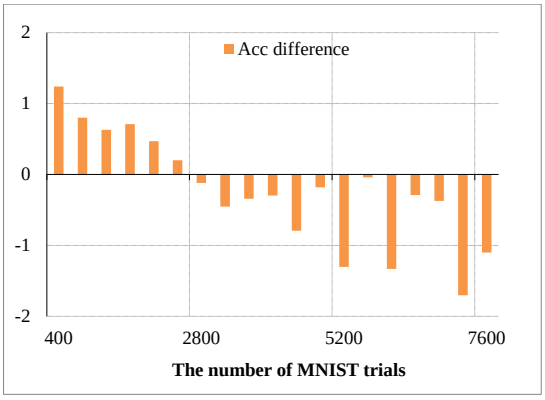

Figure 4. The difference between the two Omniglot test accuracies obtained before and after training on MNIST task.

more time for MetaNet to adapt its meta weights on the second problem so that it may forget the knowledge about Omniglot. Each run was repeated five times and we report the average statistics. For every run, the network and the optimizer were reinitialized and the training started from scratch.

In Figure 4, we plotted the accuracy difference between two Omniglot test performances obtained before and after training on the MNIST task. The performance improvement (y-axis) after training on the MNIST tasks ranges from -1.7% to 1.24% depending on the training time (x-axis). The positive values indicate that the training on the second problem automatically improves the performance of the earlier task exhibiting the reverse transfer property. Therefore, we can conclude that MetaNet successfully performs reverse transfer. At the same time, it is skilled on MNIST one-shot classification. The MNIST training accuracy reaches over 72% after 2400 MNIST trials. However, reverse transfer happens only up to a certain point in MNIST training (2400 trials). After that, the meta weights start to forget the Omniglot information. As a result from 2800 trials onwards, the Omniglot test accuracy drops. Nevertheless even after 7600 MNIST trials, at which point the MNIST training accuracy reached over 90%, the Omniglot performance drop was only 1.7%.

5. Discussion and Future Work

One-shot learning in combination with a meta learning framework can be a useful approach to address certain neural network drawbacks related to rapid generalization with small data and continual learning. We present a novel meta learning method, MetaNet, that performs a generic knowledge acquisition in a meta space and shifts the parameters and inductive biases of underlying neural networks via fast parameterization for the rapid generalization.

Under the MetaNet framework, an important consideration is the type of higher order meta information that can be extracted as a feedback from the model when operating on a new task. One desirable property here is that the meta information should be generic and problem independent. It should also be expressive enough to explain the model setting in the current task space. We explored the use of loss gradients as meta information in this work. As shown in the results, using the gradients as meta information seems to be a promising direction. MetaNet obtains state-of-the art results on several one-shot SL benchmarks and leads to a very flexible AI model. For instance, in MetaNet we can alternate between different $softmax$ layers on the fly during test. It supports continual learning up to a certain point. We observed that neural nets with fixed slow weights can perform well for new task inputs when augmented with the fast weights. When the slow weights are updated during training, it learns domain biases resulting in even better performance on identification of new concepts within the same domain. However, one could expect a higher performance from the fixed weight network when aiming for one-shot generalization across distant domains.

An interesting future direction would be in exploring a new type of meta information that is more robust and expressive, and in developing synaptic weights that are capable of maintaining such higher order information. One could take inspiration from the meta learning process in the brain and ask whether the brain operates on some kind of higher order information to generalize across tasks and acquire new skills.

The rapid parameterization approach presented here has been shown to be an effective alternative to the direct optimization methods that learn to update network parameters for one-shot generalization. However, a problem this approach poses is the integration of slow and fast weights. As a solution to this, we presented a simple layer augmentation method. Although the layer augmentation worked reasonably well, this method becomes difficult when a neural net has many types of parameters operating in multiple different time-scales. For example, a single base learner equipped with three types of weights (slow, example-specific, and task-level weights) integrated under the layer augmentation paradigm could not perform as well as a simpler one. Therefore, a potential extension would be to train MetaNet so it can discover its own augmentation schema for efficiency.

MetaNet can readily be applied to parameterize policies in reinforcement learning and imitation learning, leading to an agent with one-shot and meta learning capabilities. MetaNet based on recurrent networks as underlying learners could lead to useful applications in sequence modeling and language understanding tasks.

Acknowledgements

We would like to thank the anonymous reviewers and our colleagues, Jesse Lingeman, Abhyuday Jagannatha and John Lalor for their insightful comments and suggestions on improving the manuscript. This work was supported in part by the grant HL125089 from the National Institutes of Health and by the grant 1I01HX001457-01 supported by the Health Services Research & Development of the US Department of Veterans Affairs Investigator Initiated Research. Any opinions, findings and conclusions or recommendations expressed in this material are those of the authors and do not necessarily reflect those of the sponsor.

References

Andrychowicz, Marcin, Denil, Misha, Gomez, Sergio, Hoffman, Matthew W, Pfau, David, Schaul, Tom, and de Freitas, Nando. Learning to learn by gradient descent by gradient descent. In *Advances in Neural Information Processing Systems*, pp. 3981–3989, 2016.

Ba, Jimmy, Hinton, Geoffrey E, Mnih, Volodymyr, Leibo, Joel Z, and Ionescu, Catalin. Using fast weights to attend to the recent past. In *Advances In Neural Information Processing Systems*, pp. 4331–4339, 2016.

Bengio, Yoshua, Bengio, Samy, and Cloutier, Jocelyn. *Learning a synaptic learning rule*. Université de Montréal, Département d'informatique et de recherche opérationnelle, 1990.

Chopra, Sumit, Hadsell, Raia, and LeCun, Yann. Learning a similarity metric discriminatively, with application to face verification. In *Computer Vision and Pattern Recognition, 2005. CVPR 2005. IEEE Computer Society Conference on*, volume 1, pp. 539–546. IEEE, 2005.

David Ha, Andrew Dai and Le, Quoc V. Hypernetworks. In *ICLR 2017*, 2017.

De Brabandere, Bert, Jia, Xu, Tuytelaars, Tinne, and Van Gool, Luc. Dynamic filter networks. In *Neural Information Processing Systems (NIPS)*, 2016.

Edwards, Harrison and Storkey, Amos. Towards a neural statistician. In *ICLR 2017*, 2017.

Gomez, Faustino and Schmidhuber, Jürgen. Evolving modular fast-weight networks for control. In *International Conference on Artificial Neural Networks*, pp. 383–389. Springer, 2005.

Goodfellow, Ian J, Mirza, Mehdi, Xiao, Da, Courville, Aaron, and Bengio, Yoshua. An empirical investigation of catastrophic forgetting in gradient-based neural networks. In *ICLR 2014*, 2014.

Graves, Alex, Wayne, Greg, and Danihelka, Ivo. Neural turing machines. *arXiv preprint arXiv:1410.5401*, 2014.

Greengard, Paul. The neurobiology of slow synaptic transmission. *Science*, 294(5544):1024–1030, 2001.

Harlow, Harry F. The formation of learning sets. *Psychological review*, 56(1):51, 1949.

Hinton, Geoffrey E and Plaut, David C. Using fast weights to deblur old memories. In *Proceedings of the ninth annual conference of the Cognitive Science Society*, pp. 177–186, 1987.

Hochreiter, Sepp, Younger, A Steven, and Conwell, Peter R. Learning to learn using gradient descent. In *International Conference on Artificial Neural Networks*, pp. 87–94. Springer, 2001.

Kaiser, Lukasz, Nachum, Ofir, Roy, Aurko, and Bengio, Samy. Learning to remember rare events. In *ICLR 2017*, 2017.

Kirkpatrick, James, Pascanu, Razvan, Rabinowitz, Neil, Veness, Joel, Desjardins, Guillaume, Rusu, Andrei A, Milan, Kieran, Quan, John, Ramalho, Tiago, Grabska-Barwinska, Agnieszka, et al. Overcoming catastrophic forgetting in neural networks. *arXiv preprint arXiv:1612.00796*, 2016.

Koch, Gregory. *Siamese neural networks for one-shot image recognition*. PhD thesis, University of Toronto, 2015.

Lake, Brenden M, Salakhutdinov, Ruslan R, and Tenenbaum, Josh. One-shot learning by inverting a compositional causal process. In *Advances in neural information processing systems*, pp. 2526–2534, 2013.

Lake, Brenden M, Salakhutdinov, Ruslan, and Tenenbaum, Joshua B. Human-level concept learning through probabilistic program induction. *Science*, 350(6266):1332–1338, 2015.

Li, Ke and Malik, Jitendra. Learning to optimize. In *ICLR 2017*, 2017.

Maclaurin, Dougal, Duvenaud, David, and Adams, Ryan. Gradient-based hyperparameter optimization through reversible learning. In *International Conference on Machine Learning*, pp. 2113–2122, 2015.

Mitchell, Tom M, Thrun, Sebastian B, et al. Explanation-based neural network learning for robot control. *Advances in neural information processing systems*, pp. 287–287, 1993.

Munkhdalai, Tsendsuren and Yu, Hong. Neural semantic encoders. In *Proceedings of the 15th Conference of the European Chapter of the Association for Computational Linguistics: Volume 1, Long Papers*, pp. 397–407, Valencia, Spain, April 2017. Association for Computational Linguistics. URL http://www.aclweb.org/anthology/E17-1038.

Ravi, Sachin and Larochell, Hugo. Optimization as a model for few-shot learning. In *ICLR 2017*, 2017.

Santoro, Adam, Bartunov, Sergey, Botvinick, Matthew, Wierstra, Daan, and Lillicrap, Timothy. Meta-learning with memory-augmented neural networks. In *Proceedings of The 33rd International Conference on Machine Learning*, pp. 1842–1850, 2016.

Schmidhuber, J. *Reducing the Ratio Between Learning Complexity and Number of Time Varying Variables in Fully Recurrent Nets*, pp. 460–463. Springer London, London, 1993a. ISBN 978-1-4471-2063-6. doi: 10.1007/978-1-4471-2063-6_110. URL http://dx.doi.org/10.1007/978-1-4471-2063-6_110.

Schmidhuber, J. A neural network that embeds its own meta-levels. In *IEEE International Conference on Neural Networks*, pp. 407–412 vol.1, 1993b. doi: 10.1109/ICNN.1993.298591.

Schmidhuber, Jürgen. *Evolutionary principles in self-referential learning*. PhD thesis, Technical University of Munich, 1987.

Schmidhuber, Jürgen. Learning to control fast-weight memories: An alternative to dynamic recurrent networks. *Neural Computation*, 4(1):131–139, 1992.

Srivastava, Rupesh K, Masci, Jonathan, Kazerounian, Sohrob, Gomez, Faustino, and Schmidhuber, Jürgen. Compete to compute. In *Advances in neural information processing systems*, pp. 2310–2318, 2013.

Sukhbaatar, Sainbayar, Weston, Jason, Fergus, Rob, et al. End-to-end memory networks. In *Advances in neural information processing systems*, pp. 2440–2448, 2015.

Vilalta, Ricardo and Drissi, Youssef. A perspective view and survey of meta-learning. *Artificial Intelligence Review*, 18(2):77–95, 2002.

Vinyals, Oriol, Blundell, Charles, Lillicrap, Tim, Wierstra, Daan, et al. Matching networks for one shot learning. In *Advances in Neural Information Processing Systems*, pp. 3630–3638, 2016.

Weston, Jason, Chopra, Sumit, and Bordes, Antoine. Memory networks. In *In Proceedings Of The International Conference on Representation Learning (ICLR 2015)*, San Diego, California, May 2015. In press.

Younger, A Steven, Conwell, Peter R, and Cotter, Neil E. Fixed-weight on-line learning. *IEEE Transactions on Neural Networks*, 10(2):272–283, 1999.

A. Training Details

To train and test MetaNet on one-shot learning, we adapted the training procedure introduced by Vinyals et al. (2016). First, we split the data into training and test sets consisting of two disjoint classes. We then formulate a series of tasks (trials) from the training set. Each task has a support set of N classes with one image per, resulting in an N-way one-shot classification problem. In addition to the support set, we also include L number of labeled examples in each task set to update the parameters θ during training. For testing, we follow the same procedure to form a set of test tasks from the disjoint classes. However, now MetaNet assigns class labels to L examples based only on the labeled support set of each test task.

For the one-shot benchmarks on the Omniglot dataset, we used a CNN with 64 filters as the base learner b. This CNN has 5 convolutional layers, each of which is a 3 x 3 convolution with 64 filters, followed by a $ReLU$ non-linearity, a 2 x 2 max-pooling layer, a fully connected (FC) layer, and a softmax layer. Another CNN with the same architecture is used to define the dynamic representation learning function u, from which we take the output of the FC layer as the task dependent representation r. We trained a similar CNNs architecture with 32 filters for the experiment on Mini-ImageNet. However for computational efficiency as well as to demonstrate the flexibility of MetaNet, the last three layers of these CNN models were augmented by fast weights. For the networks d and m, we used a single-layer LSTM with 20 hidden units and a three-layer MLP with 20 hidden units and $ReLU$ non-linearity. As in Andrychowicz et al. (2016), the parameters G and Z of d and m are shared across the coordinates of the gradients ∇ and the gradients are normalized using the same preprocessing rule (with $p = 7$). The MetaNet parameters θ are optimized with ADAM. The initial learning rate was set to 10^{-3}. The model parameters θ were randomly initialized from the uniform distribution over [-0.1, 0.1).

B. MNIST as Out-Of-Domain Data

We treated MNIST images as a separate domain data. Particularly a model is trained on the Omniglot training set and evaluated on the MNIST test set in 10-way one-shot learning setup. We hypothesize that models with a high dynamic should perform well on this task.

In Figure 5, we plotted the results of this experiment. MetaNet- achieved 71.6% accuracy which was 0.6% and 3.2% lower than the other variants with fast weights. This is not surprising since MetaNet without dynamic representation learning function lacks an ability to adapt its parameters to MNIST image representations. The standard MetaNet model achieved 74.8% and MetaNet+ obtained 72.3%. Matching Net (Vinyals et al., 2016) reported 72.0% accuracy in this setup. Again we did not observe improvement with MetaNet+ model here. The best result was recently reported by using a generative model, Neural Statistician, that extends variational autoencoder to summarize input set (Edwards & Storkey, 2017).

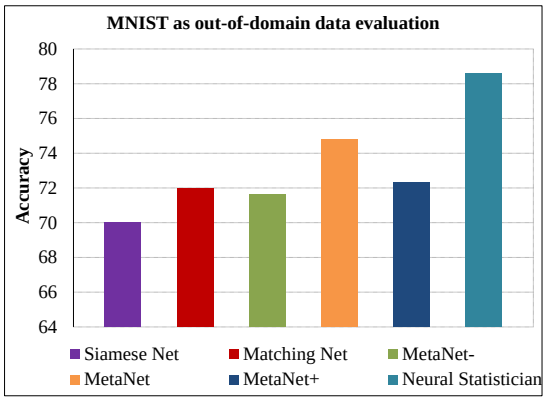

Figure 5. MNIST 10-way one-shot classification results.